IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Predicting Software Defects with Machine Learning: A New Paradigm for Enhanced Software Quality Management

[1]Akshata .S. Rane, [2]Teslin Jacob

[1]Student, [2]Professor

[1] Department of Computer Science and Engineering,
Goa College Of Engineering Goa, India

Abstract: Software defect prediction is a critical task that can make or break a software project. The frustration and disappointment that come with discovering a defect after a release can be overwhelming, especially when it could have been prevented. That's why we, as machine learning researchers, are passionate about finding ways to predict defects and improve software quality. In this study, we dive deep into the Jm1 dataset, carefully selecting features and pre-processing the data to ensure the best possible results. We train and evaluate a variety of machine learning algorithms, all with the goal of finding the best method to predict defects. It's a thrilling journey, filled with both successes and setbacks, but we are determined to find the optimal solution. Ultimately, our hope is that this research will have a real-world impact, helping software teams detect and fix defects before they cause serious problems. The idea of contributing to industrial success is a powerful motivator, and we are excited to see what the future holds for software defect prediction and machine learning.

Keywords: Software defect prediction, machine learning algorithms, decision trees, logistic regression, random forests, KNN.

I. INTRODUCTION

In today's world, where software is an essential part of our lives, the importance of software defect prediction cannot be overstated. As software systems continue to grow in size and complexity, it becomes increasingly difficult to manually detect defects, making it crucial to leverage the power of machine learning to automate the process. The cost of software defects can be staggering, both in terms of time and money. As a machine learning researcher, I have witnessed firsthand the frustration and disappointment that comes with discovering defects after a software release. It can be demoralizing to see your hard work go to waste, especially when it could have been prevented.

But there is hope. By developing accurate machine learning models to predict defects, we can significantly reduce the costs associated with software defects. We can catch defects early, when the cost of fixing them is still relatively low, and avoid the costly process of fixing them later in the software development lifecycle. The idea of improving software quality and reducing the risk of costly defects is a powerful motivator for me and for many others in the field of software engineering. By developing machine learning models that accurately predict defects, we can make a real-world impact, enabling software developers to deliver high-quality software that meets the needs of their users.

II. PROBLEM STATEMENT

Software defects are an all-too-common problem that can wreak havoc on software projects, causing delays, cost overruns, and even outright failure. These defects can take many forms, from simple coding mistakes to more complex architectural issues. But regardless of the nature of the defect, the consequences can be severe. One of the biggest risks associated with software defects is that they often go undetected until it's too late. This can be particularly problematic in the early phases of software development, where the cost of fixing a defect is relatively low. If a defect is not caught early, however, it can become more expensive to fix as the software development process progresses. This can lead to wasted resources, missed deadlines, and a decline in overall software quality. Another challenge with software defects is that they can occur at any phase of software development. Whether it's during the planning phase, design phase, or testing phase, a defect can derail the entire project if it's not caught and fixed in a timely manner. This highlights the importance of implementing effective defect detection strategies throughout the software development lifecycle.

III. LITERATURE SURVEY

Sr.No	Paper Name	Abstract	Year of publication
1	A Semantic LSTM Model for Predicting Software Defects	The authors of this study introduce Seml, a paradigm that enables developers to predict software defects. Instead of focusing on metrics such as code complexity and lines of code, the method takes advantage of deep learning and word embedding techniques to analyze the semantic and syntax of programs. It can help them save time and money by developing software that is automatically aware of any defects.	2019
2	Compressed C4.5 Models for Predicting Software Defects	The authors of the study present a compressed version of the C4.5 models used to predict software defects. Unlike other methods, which focus on metrics such as code complexity, the Seml framework takes advantage of deep learning and word embedding techniques to analyze the program's semantic and syntax.	2012
3	Early Software Defect Prediction Through Ensemble Learning: A Comparative Study	In this study, the performance of three different ensemble learner types—Boosting, Bagging, and Rotation Forest—along with the resample technique is compared to that of single learners. Our findings demonstrate that, as compared to single learners, adopting ensemble strategies, particularly those utilising Rotation Forest and the resample methodology, considerably increases accuracy across a wide range of experimental algorithms.	
4	Software Module Defect Prediction Using Class Imbalance Learning	A wide range of machine learning techniques have been investigated to forecast software module failures in order to simplify software testing and reduce testing expenses. Unfortunately, the unbalanced structure of this kind of data makes learning such a task more challenging. In order to come up with better solutions, we investigate in this work whether and how class imbalance learning approaches might help software defect prediction. We examine various class imbalance learning approaches, such as threshold moving, ensemble algorithms, and resampling strategies. AdaBoost.NC performs the best overall out of all the approaches we looked at in terms of measurements like balance, G-mean, and Area Under the Curve (AUC).	2018
5	Heterogeneous Ensemble Classification for Software Defect Prediction Based on Segmented Patterns	The early identification of software modules that are prone to defects before the actual testing process starts is a promising strategy for increasing software quality and testing effectiveness. Software engineers can more efficiently deploy their limited resources to the modules that are more prone to flaws according to the outcomes of these predictions. For the purpose of predicting software defects, a hybrid heterogeneous ensemble approach is put forth in this study. Heterogeneous ensembles are made up of a variety of classifiers with various learning bases, each of which has advantages and disadvantages. The primary goal of the suggested method is to create knowledgeable and reliable heterogeneous categorization models.	2020
6	A hybrid technique combining machine learning and optimisation approaches is used to forecast software defects.	Rapid improvements in software technology drive the expansion of many businesses. Nowadays, software-based solutions are highly prevalent in business. Because a faulty software module can harm the growth of an industry or business, developing dependable software is becoming a difficult task for any software industry. As a result, approaches for predicting software problems must be developed early on.	2018
7	Research on cross - Project software defect prediction based on transfer learning	According to the two challenges in the prediction of cross-project software defects, the distribution differences between the source project and the target project dataset and the class imbalance in the dataset, proposing a cross-project software defect prediction method based on transfer learning, named NTrA. Firstly, solving the source project data's class imbalance based on the Augmented Neighborhood Cleaning Algorithm. Secondly, the data gravity method is used to give different weights on the basis of the attribute similarity of source project and target project data. Finally, a defect prediction model is constructed by using the Trad boost algorithm. Experiments were conducted using data, come from NASA and SOFTLAB respectively, from a published PROMISE dataset.	2021
8	Using K-PCA and different kernel-based extreme learning machines, anticipate software defects.	Overfitting is also one of the most difficult problems for SDP. The authors conducted an empirical examination of these two difficulties and analysed their possible solutions in this paper. They ran 4840 trials with five different classifiers across eight NASA programmes and 14 PROMISE repository datasets.They proposed and investigated altering the kernel function of an extreme learning machine (ELM) in conjunction with kernel principal component analysis (K-PCA) and discovered better outcomes than other	2020

		standard SDP models. They employed the synthetic minority oversampling sampling methodology to handle class imbalance issues and k-fold cross-validation to avoid overfitting. They discovered that ELM-based SDP has a high receiver operating characteristic curve across 11 of 22 datasets.	
9	Convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM) hybrid model (CBIL) for software fault prediction	The software industry has made significant efforts in recent years to improve software quality in organizations. Using proactive software defect prediction will assist developers and white box testers in detecting flaws early, saving time and effort. Traditional software defect prediction models focus on traditional source code metrics such as code complexity, lines of code, and so on. These characteristics, however, do not extract the semantics of source code. In this study, we present a hybrid model termed CBIL. CBIL can forecast the parts of source code that are faulty. It generates vectors of Abstract Syntax Tree (AST) tokens from source code. Integer vectors are transformed into dense vectors through mapping and word embedding.	2021
10	Software defect prediction based on LLE AND SVM	The proposed model employs LLE to reduce the high dimensionality and noise in software datasets, followed by training an SVM model to predict software defects. The model was evaluated using various datasets from the PROMISE repository and showed promising results.	2014

IV. FLOW DIAGRAM

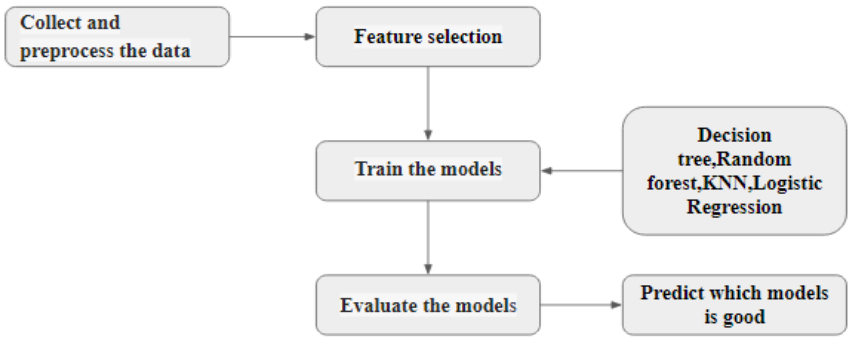

FIG 1: Flow diagram of software defects prediction

V. PROPOSED APPROACH

Dataset

The JM1 dataset is a valuable resource for researchers in the field of software defect prediction. It provides a diverse set of data from a range of application domains, making it useful for evaluating machine learning algorithms across different contexts. One interesting feature of the dataset is that it is written in "C," a programming language that is still widely used today, despite being developed in the 1970s. The data is collected using McCabe and Halstead feature extractors, which were designed to objectively characterize code features associated with software quality. While the nature of the association between these features and software quality is still under debate, the use of these features in the JM1 dataset provides a standardized and consistent basis for evaluating machine learning algorithms for software defect prediction.

Attribute Information

1. loc : numeric % McCabe's line count of code

2.v(g) : numeric % McCabe "cyclomatic complexity"

3.ev(g) : numeric % McCabe "essential complexity"

4. iv(g) : numeric % McCabe "design complexity"

5.n : numeric % Halstead total operators + operands

6. v : numeric % Halstead "volume"

7. l : numeric % Halstead "program length"

8. d : numeric % Halstead "difficulty"

9. i : numeric % Halstead "intelligence"

10.e : numeric % Halstead "effort"

11.b : numeric % Halstead

12.t : numeric % Halstead's time estimator

13.lOCode : numeric % Halstead's line count

14.lOComment : numeric % Halstead's count of lines of

comments

15.lOBlank : numeric % Halstead's count of blank lines

16.lOCodeAndComment: numeric

17.uniq_Op : numeric % unique operators

18.uniq_Opnd : numeric % unique operands

19.total_Op : numeric % total operators

20.total_Opnd : numeric % total operands 21: branchCount : numeric % of the flow graph

21.defects : {false,true} % module has/has not one or more reported defects

Machine learning algorithm

Decision Tree Classification Algorithm:

Decision Tree is a supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

Logistic Regression:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

K-Nearest Neighbour (KNN) Algorithm:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

VI. CONCLUSION

The process of software defects prediction has gained significant attention in recent years. The ability to predict and prevent bugs before they occur is highly valuable to software development teams. In this study, we explored the effectiveness of machine learning algorithms for software bug prediction. We used various metrics such as accuracy, precision, recall, F-measure, and RMSE to evaluate the performance of these algorithms. Our findings suggest that machine learning algorithms can provide accurate and reliable predictions for software bugs. However, there is still room for improvement, and future research could focus on incorporating additional software metrics into the learning process. By doing so, we may be able to enhance the accuracy of our prediction models and ultimately improve software quality

REFERENCES

1. H. Liang, Y. Yu, L. Jiang and Z. Xian, "Smell: A Semantic LSTM Model for Software Defect Prediction," in IEEE Access, vol. 7, pp. 83812-83824, 2019, doi: 10.1109/ACCESS.2019.2925313.
2. J. Wang, B. Shen and Y. Chen, "Compressed C4.5 Models for Software Defect Prediction," 2012 12th International Conference on Quality Software, Xi'an, China, 2012, pp. 13-16, doi: 10.1109/QSIC.2012.19.

3. Sayed, Ashraf & Ramadan, Nagy. (2018). Early Prediction of Software Defect using Ensemble Learning: A Comparative Study. International Journal of Computer Applications. 179. 29-40. 10.5120/ijca2018917185.

4. Divya Tomar, Sonali Agarwal, "Prediction of Defective Software Modules Using Class Imbalance Learning", *Applied Computational Intelligence and Soft Computing*, vol. 2016, Article ID 7658207, 12 pages, 2016. https://doi.org/10.1155/2016/7658207

5. Alsawalqah, H.; Hijazi, N.; Eshtay, M.; Faris, H.; Radaideh, A.A.; Aljarah, I.; Alshamaileh, Y. Software Defect Prediction Using Heterogeneous Ensemble Classification Based on Segmented Patterns. *Appl. Sci.* 2020, *10*, 1745. https://doi.org/10.3390/app10051745

6. Manjula, C. and Lilly Florence. "Hybrid Approach for Software Defect Prediction Using Machine Learning with Optimization Technique." *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering* 12 (2018): 28-32.

7. Vikas Suhag, Anchal Garg, S. K. Dubey, B. K. Sharma. 2020. Analytical Approach to Cross Project Defect Prediction. Soft Computing: Theories and Applications, 713-736.

8. Pandey, Sushant & Rathee, Deevashwer & Tripathi, Anil. (2020). Software defect prediction using K-PCA and various kernel-based extreme learning machine: an empirical study. IET Software. 14. 10.1049/iet-sen.2020.0119.

9. Farid AB, Fathy EM, Sharaf Eldin A, Abd-Elmegid LA. 2021. Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM) *PeerJ Computer Science* 7:e739 https://doi.org/10.7717/peerj-cs.739.

10. https://www.openml.org/d/1053